Time Format Keyword Proposal

Background:

There is currently a problem with the existing PDS label that is making the development of generic software for the access and graphic display of time ordered data sets difficult at best. There is a standard value for the DATA_TYPE keyword that is TIME but this value means PDS formatted time. There is no requirement that time tags in data files be formatted in one of the PDS approved formats and many (if not most) of the time-ordered data sets do not use a PDS formatted time tag. As a result, the time tag within these data sets is not described in a machine-readable form. Labels currently use a combination of the NAME and DESCRIPTION fields to describe to users the format if time columns within these files.

Proposal:

PPI is proposing a new set of keywords for use in data file labels that are used to describe the format of time columns (character strings). The primary new keyword is TIME_FORMAT. This keyword is used to modify the meaning of the keyword-value pair DATA_TYPE = TIME. It can only be used in data element (column, field) definitions where the DATA_TYPE is set to TIME. If the TIME_FORMAT keyword does not appear in the data element object, then DATA_TYPE=TIME will have the default interpretation of time in PDS standard time format. This caveat provides backward compatibility with existing labels. This proposal does not alter in any way the requirement that catalog data (templates) still require time in PDS standard format. PDS is fully justified in prescribing the format of metadata.

The value assigned to the TIME_FORMAT keyword is a format specification. This specification defines the interpretation of every character (alpha, numeric, and non alpha-numeric) included in a date/time string. Some characters are interpreted as components of the date/time while others are delimiters. Any character or group of characters that are interpreted are described using a date/time element “token”. The set of valid tokens forms a standard value set, described below. The format specification describes the parsing algorithm for the date/time string of the length BYTES (not including any quotes) from left to right.

Two additional keywords are necessary to fully describe all of the time tags thus far encountered by the PDS. These are REFERENCE_TIME and TIME_SYSTEM. Many data files do not contain a complete date/time word. Instead, they contain a time offset from some reference time. Examples include things like decimal days after 1900-01-01 00:00:00.000 UTC, or decimal seconds after 1970-01-01 00:00:00.000 UTC. The REFERENCE_TIME keyword is used to specify an offset-time origin in UTC. The REFERENCE_TIME value is always constructed using one of the two PDS-approved time formats (year-month-day or year-doy). The TIME_SYSTEM keyword is used to specify the time system used in the data file. The default time system for all PDS data is Universal Time Coordinated (UTC). If the TIME_SYSTEM keyword is not present in a COLUMN or FIELD object where the DATA_TYPE = TIME, then TIME_SYSTEM = “UTC” is implied. This default value is required for backward compatibility with previous PDS standards. Other possible time systems include Terrestrial Dynamical Time (TDT), Barycentric Dynamical Time (TDB), etc. A complete list of TIME_SYSTEM standard values is included in the PDS Data Dictionary.

The label construct will go something like:

```
OBJECT              = COLUMN
  NAME              = "START TIME"
  DATA_TYPE         = TIME
  TIME_FORMAT       = "format-specification”
  START_BYTE        = 1
  BYTES             = 23
```

.
.
.
END_OBJECT = COLUMN

This proposal contains a set of tokens that can be used to create a format specification date/time string. Any character within the specified element (column) that is not explicitly described by a token is interpreted as a literal character. Floating-point forms are terminated either by a non-numeric character or an end of the field (out of bytes). Any precision can be specified. Any date/time element that is not specified is assumed to be zero. However by not specifying lower order date/time elements a precision is implied. The % character is used as the token delimiter at both ends (start/end) of the proposed tokens. The parsing of the time format string by software will be made easier by using both leading and trailing delimiters.

There are two classes of tokens (specific and generic) that are required to provide the necessary generality. Specific tokens describe values that are themselves part of a standard value set. Examples are tokens that describing things like "month", "day of month", "hour of day", etc. These values are typically character strings or non-negative integers. Generic tokens are used to describe relative time or date elements that do not lend themselves to such rigid definitions. Generic tokens are used to describe numeric time/date elements that may be positive or negative and whose reference origin may not be a traditional date/time origin. Examples of time elements that must be described by a generic token are 'seconds after periapsis', 'fractional seconds', 'fractional days since launch', etc. A generic token describes a value that can either be referenced to another value (next largest date/time token) within the TIME_FORMAT string, or to an origin defined by using the REFERENCE_TIME keyword. This keyword is used to provide a zero reference for generic date/time tokens when none is provided within the data file. The reference 'time' within the REFERENCE_TIME keyword must be stated to implied precision of the token it modifies. REFERENCE_TIME values are given in one of the two the PDS standard date/time formats (year-month-day or year-doy).

The time tag in many data files is not written as a string or using a write statement that forces leading zeros in numeric fields. In this specification, all numeric values are required to be right justified. Any leading spaces are interpreted as zeros and trailing spaces that are not explicitly identified in the format specification are considered to be a specification error. The BYTES keyword in the COLUMN object specifies the input string-length to be parsed. All specific tokens expect substrings of the specified string length. Fields described by generic tokens (numeric only) have no predetermined string-length. These substrings are terminated either by a non-numeric character or by reaching the end of the input string (BYTES).

The selection of tokens used to describe a time word is non-unique. This is not a problem or concern. As long as interpretation of the time format tokens results in a unique time value the time word is properly described. Both the time format tokens and the elements of a time string that they identify are explicitly case insensitive. Users should not attribute any particular meaning to the use of all uppercase in the token specification below. Spacecraft clock formats are specifically excluded from this usage. Spacecraft clock values already have a set of descriptive keywords.

Each of the tokens are described as follows:

%YEAR% is the four-digit integer year token (length = 4).

%YR% is the two-digit integer year token. Values greater then 50 are referenced to 1900 and values less or equal to 50 are referenced to 2000 unless modified by an REFERENCE_TIME keyword (length = 2).

%YEARBC% is the four-digit integer year BC token (length = 2).

%FYEAR% is the generic fractional year token. There can be any number of digits before the decimal point, however, four should be sufficient for most data sets. If less than 4 digits precede the decimal point, leading zeros are implied. This token may be used to indicate fractional years since some

starting date/time by using the REFERENCE_TIME keyword to specify the reference date/time. If the REFERENCE_TIME token is not used, FYEAR is referenced to 0001 AD for positive values and negative values indicate years BC.

%MM% is the two-digit integer month token. Valid range is 1-12 (length = 2).

%MON% is the three character month abbreviation token from the set {**JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC**}.The set is not case sensitive so that **Jan = jan = JAN** (length=3).

%MONTH% is the full month name (English) token where month is from the set {**January, February, March, April, May, June, July, August, September, October, November, December**}. The month value is not case sensitive so that **May = may = MAY** (length = 3-8).

%DD% is the two-digit integer day of month token. Valid range is 1-31 (length = 2).

%DOY% is a three-digit integer day of year token where Jan 1 = 001. Valid range is 1-366 (length=3)

%DOY0% is a three-digit integer day of year token where Jan 1 = 000. If less than three digits are found then leading zero's are implied. Valid range is 0-365 (length = 3).

%FDAY% is the generic fractional day token. The field is terminated by a non-numeric character when the column BYTES is reached. FDAY can be either floating point or integer and can be either positive or negative. The start (FDAY=0.0) date/time for the generic FDAY counter can be specified by a year elsewhere in the TIME_FORMAT string or by using the REFERENCE_TIME keyword. Julian dates (absolute and relative) are specified by using the FDAY token and a REFERENCE_TIME keyword that specifies the reference date/time FDAY can be used with TIME_FORMAT tokens that specify time of day.

%HR% is the two digit integer hour of day token. If only one digit is found, then a leading zero is implied. Valid range is 0-23 (length = 2).

%FHR% is the generic fractional hour token. The field is terminated by a non-numeric character or when the column BYTES is reached. FHR can be either floating point or integer and can be either positive or negative. The start date/time for the generic FHR can be set elsewhere in the TIME_FORMAT string by the next largest time element token (default) or by using the REFERENCE_TIME keyword to specify the reference date/time.

%MIN% is the two-digit integer minute of hour token. If only one digit is found, then a leading zero is implied. Valid range is 0-59 (length = 2).

%FMIN% is the generic fractional minute token. The field is terminated by a non-numeric character or when the column BYTES is reached. FMIN can be either floating point or integer and can be either positive or negative. The start date/time for the generic FMIN can be set elsewhere in the TIME_FORMAT string by the next largest time element token (default) or by using the REFERENCE_TIME keyword to specify the reference date/time.

%SEC% is the two-digit second of minute token. If only one digit is found, then a leading zero is implied. The valid range is 0-60 (length = 2).

%FSEC% is the generic fractional second token. The field is terminated by a non-numeric character or when the column BYTES is reached. FSEC can be either floating point or integer and can be either positive or negative. The start date/time for the generic FSEC is set by the next largest time element token in the TIME_FORMAT string.

%ESEC% is the total elapsed fractional seconds (including leapseconds) since some reference time. Values can be positive, negative, or zero. The REFERENCE_ TIME keyword is used to specify the date/time origin.

%USEC% is the elapsed fractional seconds (no leapseconds) since some reference time. Many Unix systems use fractional seconds since 1970 as the reference time. These systems have no knowledge of leapseconds. Values can be positive, negative, or zero. The REFERENCE_ TIME keyword is used to specify the date/time origin.

%MSEC% is the three-digit integer millisecond of second token. If less than 3 digits are found, the leading zeros are implied. The valid range is 0-999 (length = 3).

%FMSEC% is the generic fractional millisecond token. The field is terminated by a non-numeric character or when the column BYTES is reached. FMSEC can be either floating point or integer and can be either positive or negative. The start date/time for the generic FMSEC is set by the next largest time element token in the TIME_FORMAT string.

Usage Examples:

All representations given below are for the same time.

1994-08-17T03:31:27.400Z PDS standard formatted time

```
TIME_FORMAT = "%YEAR%-%MM%-%DD%T%HR%:%MIN%:%SEC%.%MSEC%Z"
            = "%YEAR%-%MM%-%DD%T%HR%:%MIN%:%FSEC%Z"
```

August 17, 1994 03:31:27.400

```
TIME_FORMAT = "%MONTH% %DD%, %YEAR% %HR%:%MIN%:%SEC%.%MSEC%"
```

94229 12687.4 Standard date/time form used by PVO mission

```
TIME_FORMAT = "%YR%%DOY% %FSEC%"
```

903324087.400 Commonly called Clinetime after Neal Cline

```
TIME_FORMAT          = "%USEC%
REFERENCE_ TIME      = 1966-01-01T00:00:00.000    1 millisecond precision implied
```

1994 228.14684 FDAY takes origin to be 1984-01-01 00:00:00 UT

```
TIME_FORMAT = "%YEAR% %FDAY%    Value above has 1 second precision
```

228.14684 Year omitted in data file, reference time required

```
TIME_FORMAT          = "%FDAY%
REFERENCE_ TIME      = 1994-01-01T00:00:00    1 second precision implied
```

Impact:

The primary impacts of adopting proposal will be in the software and documentation area. Software that are may be impacted includes TBTOOL, NASAVIEW, and the label/volume verifier software (VV, SLVTOOL, TOOLD). Since this software does not affect the interpretation of time in catalog templates, there should be **NO IMPACT ON THE CATALOG LOADER OR DATA INGESTION**.

TBTOOL - will complain that the label contains an unknown keyword but will still allow users to view table contents. This response has been verified.

NASAVIEW - same response as TBTOOL (verified). However, if NASAVIEW is upgraded to provide a graphics capability for tables (already supported for tables generated from NIMS cubes) then it would be natural to allow users to plot tabular data vs. time that would require a parsing capability for time formats.

VV - The label verifier component of this software (SLVTOOL) should be modified to validate the usage of this keyword and it's valid values. I expect that w/o modification the software will simply complain about

an unknown keyword (repeatedly!) and then fail to validate the proper formation of the value.

DOCUMENTATION - Time and time formats are discussed throughout the various PDS standards documents (STANDARDS REF, DATA PREPARER'S WORKBOOK, DATA DICTIONARY). All of these documents would have to be updated to reflect the impact of this new keyword/value pair.

L^3 - The easiest way to implement the changes in the various PDS tools is to update the underlying library. The parsing of TIME_FORMAT values should be added to L^3.